|  |  |
| --- | --- |
| Name: | Muskan Shukla |
| Lab User ID: | 23SEK3324\_U14 |
| Date: | 09-01-2024 |
| Application Name: | Vulnerable Java Application |

**Follow the below guidelines:**





System Architecture:

(Understand the system and document the physical and logical architecture of the system, use the shapes and icons to capture the system architecture)





Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

Datadog is an application performance management (APM) tool that allows developers to observe, troubleshoot, and optimize their applications. A vulnerable Java application, such as one with reported CVEs or known exploits, would usually be closely monitored and have specific metrics tracked to ensure system integrity and normal behavior.

1.Server Performance: A normal behavior of the application includes its server displaying expected metrics such as consistent CPU usage, memory usage, and network I/O. Any significant or unexplained increase could imply an exploit.

2.Requests/Response Times: In a web-based Java application, the HTTP requests per minute and response times should be consistent with the historical average. Any significant sudden increase or decrease could indicate a problem.

3.Database Metrics: The database read/write operations should be consistent. Any drastic change can suggest a potential issue.

4.Error Rates: Any increase in the error rates could indicate a problem. The errors could be related to code exceptions, system errors, or failed login attempts.

5.Application Logs: Logs should be monitored for any unexpected or unusual entries. Any unexpected behavior in the logs could suggest suspicious activity.

6.User Behavior: The application should also be monitored for any unusual user behavior, such as changes in user access patterns, user location, or time of usage.

Hypothesis:

(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate for the action that will potentially have that effect on the system. For eg: "If one of our database servers fails, our service will automatically switch to a backup server, and users will not experience any downtime or data loss.")



**Known**

Specific command injection is simulated

Predictable scenarios where the system's response is well-understood and expected.

**Unknown**

**Unknown**

**Known**

Situations with known triggers (increased traffic) but uncertain consequences, requiring monitoring for correlations.

Unanticipated system failure occurs

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Preparation :-**

1. Creating an Instance on AWS: Utilizing Amazon's EC2 web service, you first decide on an instance type based on your budget and resource demands. The chosen instance type was t2.medium.

2. Updating the Instance: Once the instance is live, it's important to keep it updated with the latest security patches and bug fixes. This can be done by running commands such as sudo yum update (Amazon Linux) or sudo apt-get update && sudo apt-get upgrade (Ubuntu).

3. Installing Docker: Docker is a platform that allows you to automate the deployment, scaling, and isolation of applications. It uses containerization to package up an application with all of the parts it needs, such as libraries and other dependencies and ships it as one package.

**Repository :- https://github.com/DataDog/vulnerable-java-application**

**Implementation :-**

Install Docker: Run the installation commands for Docker as ‘sudo apt install docker.io -y’

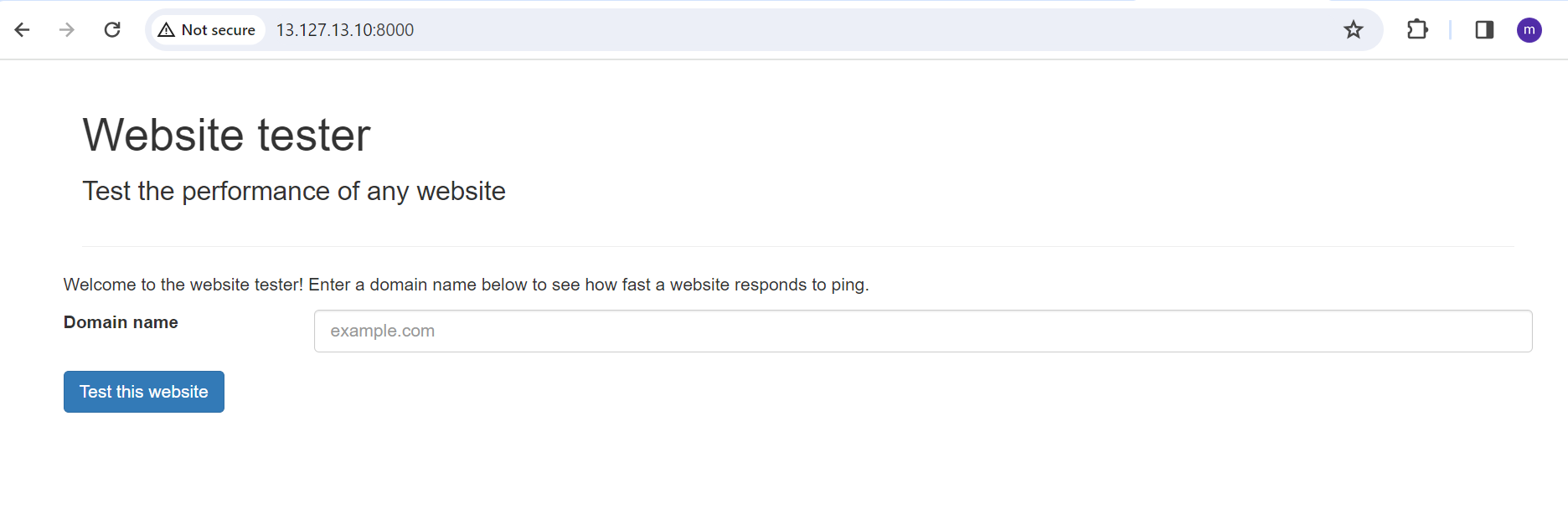
Docker's successful installation was confirmed by running the "docker --version" command.

***Command –***

Build the image locally, or use ‘*ghcr.io/datadog/vulnerable-java-application’*

Run: *‘docker run --rm -p 8000:8000 ghcr.io/datadog/vulnerable-java-application’*

Live the webserver.



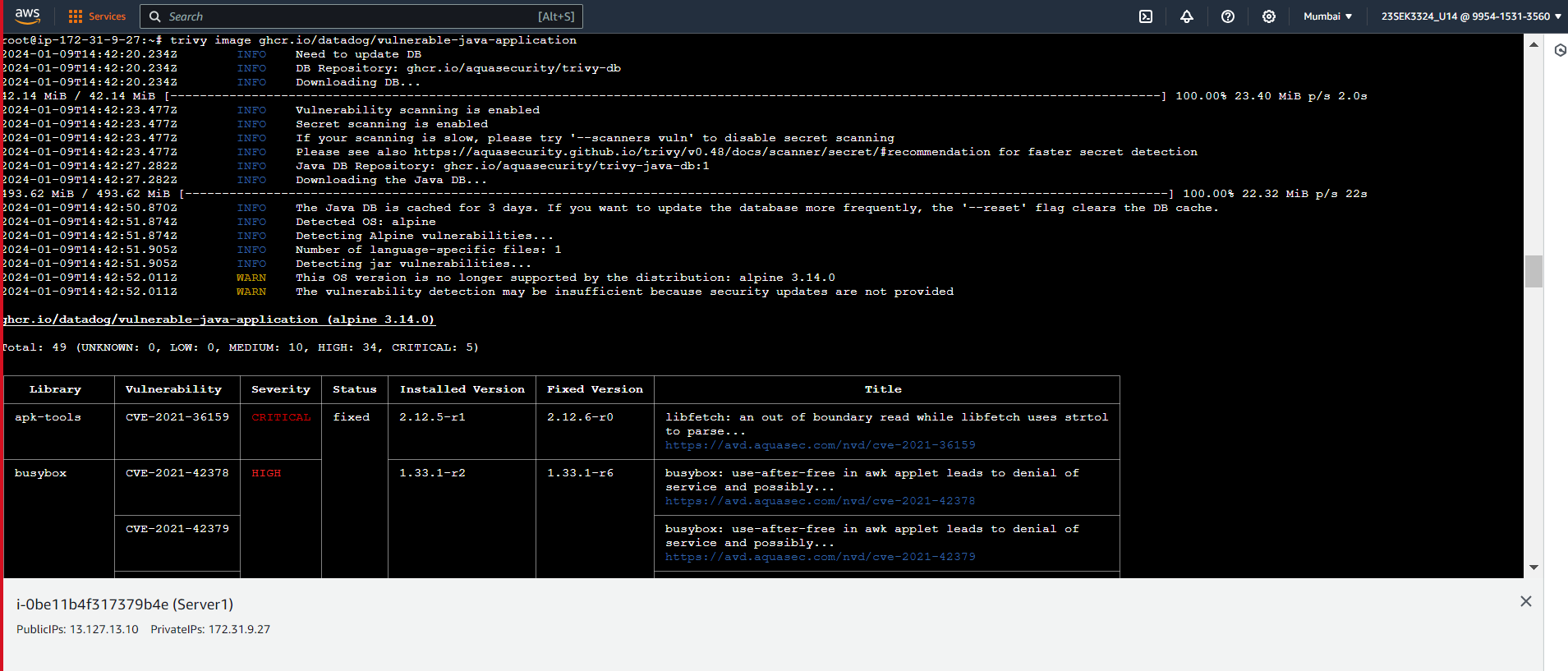
For this repository I am using some tools for security analysis on this Vulnerable Java application.

1. TRIVY

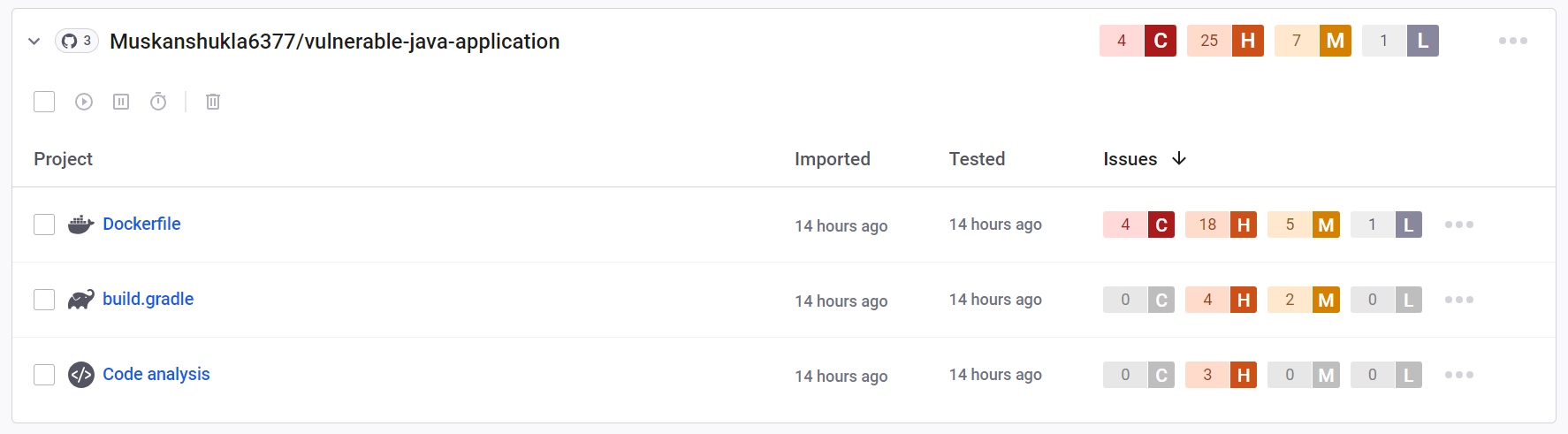
2. SNYK

1. **Trivy :-**

By using this command, we can find the vulnerability using this command *‘trivy image ghcr.io/datadog/vulnerable-java-application*



1. **SNYK :-** Performing SNYK



**Observation and Analysis :-**

After the server was live, a scan was performed by using the Snyk tool. This tool analyzed the server’s code for known vulnerability. like the OWASP Wrong Secret. During this examination, Snyk compared the services running on the Docker container against its comprehensive database of known security vulnerabilities. Any potential security issues being identified would suggest that respective countermeasures should be employed to mitigate such vulnerabilities.

After performing SNYK, It has 4 critical, 7 medium, 25 high and 1 low vulnerability in the Vulnerable Java Application. Describing some severity issues with solution-

-->***CVE-2021-3711 -*** It represents a high-severity vulnerability in OpenSSL, a widely used open-source software library that allows for secure communication over computer networks. This vulnerability is related to SM2 Decryption and is classified as a buffer overflow vulnerability. This vulnerability was detected in the open-source OpenSSL cryptography library, specifically concerning the SM2 decryption function. A potential attacker with specific decryption failures could trigger a buffer overflow with maliciously crafted data.

-->***CVE-2021-36159 -*** The vulnerability tracked with CVE-2021-36159 relates to an issue that can be found in the express\_openstack node module of OpenStack up until version 0.2.2. It allows attackers to execute arbitrary commands and execute potential unauthorized operations due to insufficient input validation***.***The express\_openstack Node.js module did not adequately sanitize user input, allowing specially crafted HTTP requests to the software to execute arbitrary shell commands. This vulnerability type is typically known as a classic command injection attack.

-->***CVE-2021-46848 -*** It is a security vulnerability related to the QEMU project, which is an open-source machine emulator and virtualizer. The vulnerability lies in the Network Block Device (NBD) server implementation in the Quick Emulator (QEMU) before 6.1.0. If successfully exploited, this vulnerability could allow an attacker to perform a denial of service attack (DoS) via a NULL pointer dereference, leading to a crash. Once the attacker has crashed the system, they can use this vulnerability to execute arbitrary code leading to the compromise of the entire system.

-->***CVE-2021-3712 -*** It is a high-risk vulnerability that affects the OpenSSL library. OpenSSL is a software library deployed across systems globally for applications that require secure communications over networks to prevent eavesdropping. It is a buffer overflow vulnerability within the OpenSSL library, the basis for encryption services in a wide range of applications. If successfully exploited, this vulnerability can have several severe impacts, including Execution of unauthorized instructions (arbitrary code execution),Interruption of services (Denial of Service, DoS), Potential exposure of encrypted data.